MULTI-PATH DATA RETRIEVAL FROM REDUNDANT ARRAY

CROSS-REFERENCE TO RELATED APPLICATIONS

[01] The present application is related to Patent Application Serial No. (Attorney Docket No. ARC9-2003-0014-US1), entitled "Anamorphic Codes", Patent Application Serial No. (Attorney Docket No. ARC9-2003-0015-US1), entitled "Autonomic Parity Exchange," and Patent Application Serial No. (Attorney Docket No. ARC9-2003-0040-US1), entitled "RAID 3 + 3" each co-pending, co-assigned and filed concurrently herewith, and each incorporated by reference herein. The present application is also related to co-pending and co-assigned Patent Application Serial No. (Attorney Docket No. YOR9-2003-0069-US1), which is also incorporated by reference herein.

BACKGROUND OF THE INVENTION

Field of the Invention

[02] The present invention relates to the field of data storage systems. In particular, the present invention relates a system and a method for determining a best pathway to requested data based on a metric, thereby improving device failure protection of an array of storage devices.

Description of the Related Art

[03] A conventional array of storage devices typically has sufficient redundancy so that when a storage device fails, information contained in the failed storage device can be reconstructed from the remaining storage devices. See, for example, U.S. Patent No.

5,579,475 to M.M. Blaum et al., entitled "Method and Means for Encoding and Rebuilding the Data Contents of Up to Two Unavailable DASDs in a DASD Array Using Simple Non-Recursive Diagonal and Row-Parity," which discloses the operation of an array having distance D = 3. See also, N.K. Ouchi, "Two-Level DASD Failure Recovery Method," IBM Technical Disclosure Bulletin Vol. 36, 03 Mar 1993, discloses the operation required for reconstructing data from an array with failures and having distance D = 3.

[04] Redundancy may also be used for improving performance. See, for example, E.J Schwabe et al., "Evaluating Approximately Balanced Parity-Declustered Data Layouts for Disk Arrays," ACM 0-89791-813-4/96/05 1996, which disclose data layouts for efficient positioning of redundant information for performance. See also G.A. Alvarez et al., who, in "Tolerating Multiple Failures in RAID Architectures," ACM 0-89791-901-7/97/0006 1997, disclose properties and construction of a general multiple-parity array using 8-bit finite fields, and L. Xu and J. Bruck, who, in "Improving the Performance of Data Servers Using Array Codes," Paradise ETR027 (CalTech) 1998, describe the use of a maximum distance separation (MDS) code for improving system response.

[05] Existing RAID (Redundant Array of Independent Disks) systems have multiple pathways, or routes, for reading requested data. Often, however, there are only a few available pathways, of which only one pathway is efficient. For example, a RAID 5 system provides two pathways for reading information. One pathway is by directly reading the sector containing the requested information. The second pathway is by reconstructing the data sector containing the requested information by reading the appropriate sector from each other storage unit in the RAID 5 array. As another example, an N storage unit array that is configured as a RAID 6 system can read a data sector directly or can read all other storage units, except one. Accordingly, there are N ways for RAID 6 to read a data sector. In both instances, there is one efficient pathway and one or more inefficient pathways.

[06] Thus, when there are many pathways for obtaining requested data, for example, more pathways than the number of storage devices in the array, it is not trivially clear which pathway provides in the highest performance for a storage system. Consequently, what are needed are a system and a method for determining which pathway to select when a storage system uses a redundancy method having many pathways to requested data.

BRIEF SUMMARY OF THE INVENTION

[07] The present invention provides a system and a method for determining which pathway to select when a storage system uses a redundancy method having many pathways to requested data.

[08] The advantages of the present invention are provided by a pathway determination system for a data storage system having N storage devices and more than N pathways for retrieving requested data from the data storage system. The invention also permits at least one of the storage devices to be a failed storage device. The pathway determination system includes a sorter, an assigner and a collector. The sorter receives a read request and separates the read request into an appropriate segment size for sending to the storage devices of the data storage system. The assigner includes a permutation generator that generates a set of read permutations satisfying the read request, and a cost calculator that calculates an expense of each permutation based on a metric calculated from performance information received from the storage devices of the storage system. The read permutations can be generated after the read

request is received. Alternatively, the read permutations can be precalculated before a read request is received, based on the architecture of the storage system. The cost calculator uses queue length information and estimated current cost information to assign a value based on a metric to the read permutations. Additional information on the costs are passed back to the permutation generator as hints for the permutation generator to reduce the number of permutations it generates. The assigner selects a read permutation from the set of read permutations. The selection of read permutation is based on the value that the cost calculator assigned it. The metric used by the cost calculator is based on the anticipated resource use of the permutation,, such as on a current workload balance for the storage devices of the data system, an estimated delay before the requested data can be retrieved from the storage devices of the storage system, a number of outstanding requests in the queue of a storage device of the storage system, and/or a total queue for all outstanding requests that have been received by the storage system. The assigner sends the selected read permutation to the storage devices of the storage system. The collector receives the requested data from the N storage devices in response to the selected read permutation being sent to the storage devices. The metric can be dynamically changed based on a change in operating conditions of the storage system. Alternatively, the metric can be periodically changed based on operating conditions of the storage system.

[09] The present invention also provides a method for determining a pathway for obtaining data stored in a data storage system having N storage devices and more than N pathways for retrieving requested data from the data storage system. The storage system can include at least one failed storage device. According to the present invention, a read request is received and separated into an appropriate segment and size for sending to the storage devices of the data storage system. A set of read permutations satisfying the received read request are generated either at the time the

read request is received or in advance. A read permutation is selected from a set of read permutations based on a metric. The metric can be based on a current workload balance for the storage devices of the data system, an estimated delay before the requested data can be retrieved from the storage devices of the storage system, a number of outstanding requests in the queue of a storage device of the storage system, and/or a total queue for all outstanding requests that have been received by the storage system. An expense for each permutation is calculated based on the metric and based on performance information received from the storage devices of the storage system. Queue length information and estimated current cost information is generated and used for generating a reduced number of read permutations based on the queue length information and the estimated current cost information. The selected read permutation is sent to the storage devices of the storage system. The requested data is received from the N storage devices in response to the selected read permutation being sent to the storage devices, and the satisfied read request is returned to the requester. The metric can be dynamically changed based on a change in operating conditions of the storage system. Alternatively, the metric can be periodically changed based on operating conditions of the storage system.

BRIEF DESCRIPTION OF THE DRAWINGS

[10] The present invention is illustrated by way of example and not by limitation in the accompanying figures in which like reference numerals indicate similar elements and in which:

[11] Figure 1 shows a functional block diagram of an exemplary embodiment of a system for determining the best pathway for obtaining requested data based on a defined metric according to the present invention;

[12] Figure 2 is a functional block diagram showing greater detail of an assigner, shown in Figure 1, according to the present invention;

[13] Figure 3 depicts allowed permutations generated by a permutation generator, shown in Figure 2, for an exemplary SOLD triple according to the present invention;

[14] Figure 4 is a functional block diagram showing greater detail of a cost calculator, shown in Figure 2, according to the present invention; and

[15] Figure 5 is a functional block diagram showing greater detail of a combiner, shown in Figure 1, according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[16] The techniques of the present invention are applicable to a RAID-type system in which there are more than N pathways for returning a requested data sector, such that N is the number of storage devices in the system. Arbitrary read requests received from a host system are separated into appropriate segments and sizes for dispatch to the data storage devices of the system. The various pathways to the requested data are evaluated and the best permutation of the read request is selected and issued to the storage device array. After the selected permuted read has been gathered, the requested information is determined and returned to the host controller.

[17] The best pathway to the requested data is determined based on a metric, such as the current workload balance for the storage devices in the system, an estimated delay before the requested data can be retrieved, the number of outstanding requests in the queue of a storage device, and/or the total queue for all outstanding requests that have been accepted. The metric may be predetermined or can be continually

reevaluated and dynamically changed, such as by retroactively altering the queues of the storage devices of the system, so that the best pathway is selected as circumstances and operating conditions change. When one or more storage devices in the system are unavailable, the present invention selects the best pathway from the remaining available pathways for obtaining requested data. Information regarding the metrics that are used for selecting the best pathway can be made available externally to the storage system for evaluation by a maintenance entity.

[18] Figure 1 shows a functional block diagram of an exemplary embodiment of a system 100 for determining the best pathway for obtaining requested data based on a defined metric according to the present invention. System 100 is contained within a storage system (not shown) using a 3 data + 3 parity MDS (maximum distance separation) erasure code. A primary aspect of a storage system is to process read requests from a host controller (not shown), dispatch the commands to the storage devices of the system, and then return the retrieved data to the host controller. A 3 + 3 MDS erasure code configuration allows three disk failures without loss of data; while providing that data from any sector on a data disk can be reconstructed by combining the data contained in the appropriate sector from any three of the other disks. Accordingly, the present invention can be adapted to suit other redundancy schemes, such as RAID 51, other product codes and larger MDS codes.

[19] System 100 includes a sorter 106, an assigner 108, an array of storage devices 110-120, and a combiner 122. Storage devices 110-120 will also be respectively referred to herein as disks A, B, C, which contain data, and P, Q and R, which contain parity information. While storage devices 110-120 are illustratively shown in Figure 1 as disk drives A, B, C, P, Q and R, respectively, other mass storage devices, such as

Random Access Memory (RAM) storage devices, optical storage device, and tape storage devices, can be used as storage devices 110-120.

[20] A read request 102 that is received from the storage system (not shown), in which system 100 is contained, is passed to sorter 106. Sorter 106 sorts each read request 102 in a well-known manner into a Segment (that is, a region of the array of storage devices 110-120 corresponding to the block address of the requested data), an Offset within the segment, a Length, and a Disk (i.e., data disk A, B or C), collectively referred to herein by the acronym SOLD. Each SOLD is then passed from sorter 106 to assigner 108. Assigner 108 selects the best pathway to the data identified by the SOLD. Assigner 108 also passes determination for satisfying a SOLD to combiner 122 as a state update information 124. Combiner 122 reconstructs the data identified by the SOLD from the results received from storage devices 110-120, and a satisfied request 126 is passed back to the storage system that is external to system 100. Combiner 122 also provides feedback information 128 to assigner 108.

[21] Figure 2 is a functional block diagram showing greater detail of assigner 108, shown in Figure 1. Assigner 108 includes SOLD queues 204-208 for the data disks, a SOLD gatherer 210, a permutation generator 212, a cost calculator 216, a permutation multiplexer 218 and storage device queues 220-230. Storage device requests 202a-202c for data disks A, B and C that are received from sorter 106 (Figure 1) respectively enter random-access SOLD queues 204-208. SOLD queues 204-208 are configured as well-known I/O queues, and may combine known Quality-of-Service (QoS) features, such as windowing and elevator sorting. SOLDs that are within the same segment are removed from queues 204-208 by a SOLD gatherer 210, which then combines the removed SOLD to form a map of requests for storage devices 110, 112 and 114 (i.e., disks A, B and C) within that segment. The combined SOLDs that

are within the same segment are referred to herein as a "SOLD triple". A permutation generator 212 examines each SOLD triple and generates the allowed permutations for each received read request that satisfies the SOLD triple. The generated permutations are passed to a cost calculator 216 that determines the expense of each permutation based on a defined metric. The least expensive permutation is selected by permutation multiplexer 218 based on the defined metric. The selected permutation is passed to queues 220-230 for the storage devices 110-120, respectively. State update information 124 about the selected permutation is passed forward from permutation multiplexer 218 to combiner 122 (Figure 1). Information 232a-232f relating respectively to queue lengths and estimated current costs of queues 220-230 are passed back from storage device queues 220-230 to cost calculator 216. The cost of a queue is the amount that the cost metric is increased by placing the chosen request on that queue (one of its relationships is to the length of the queue). Additionally, queue length and estimated current cost information 232 is filtered by cost calculator 216 and passed to permutation generator 212 as hinting information 234. Hinting information 234 is used by permutation generator 212 to reduce the number of permutations that must be supplied. Performance information from storage devices 110-120 (Figure 1) is supplied as feedback information 128 to cost calculator 216 for refining the costing analysis.

[22] Figure 3 depicts allowed permutations generated by permutation generator 212 for an exemplary SOLD triple 302. SOLD triple 302 depicts a solitary read request on disk A and no read requests on disks B and C. Permutation generator 212, in the absence of any hinting information 234, produces all allowed permutations 310-330 over disks A, B, C, P, Q and R that satisfies SOLD triple 302. In particular, permutation 310 represents a single disk A read 310. Additionally, all combinations of three-disk reads of the six available disks A, B, C, P, Q and R for reconstructing the request on disk A

are shown. More specifically, permutation 312 represents disk reads of disks C, P and R. Permutation 314 represents disk reads of disks B, C and Q. Permutation 316 represents disk reads of disks B, P and R. Permutation 318 represents disk reads of disks B, C and R. Permutation 320 represents disk reads of disks B, C and P. Permutation 322 represents disk reads of disks B, Q and R. Permutation 324 represents disk reads of disks C, P and Q. Permutation 326 represents disk reads of disks B, P and Q. Permutation 328 represents disk reads of disks P, Q and R. Lastly, permutation 330 represents disk reads of disks C, Q and R. Each request, 312-330, is able to reconstruct the SOLD requested 302 by way of the design of the parity calculation.

[23] Hinting is used by permutation generator 212 for eliminating disk read combinations. For example, in a situation in which hinting provides information that disks C, P and Q are preferred because disks A and B are heavily loaded and disk R has failed, permutation generator 212 would only generate permutations 310 and 324. The process of hinting informs the permutation generator 212 which disks would be prohibitively expensive to use.

[24] Figure 4 is a functional block diagram showing greater detail of cost calculator 216, shown in Figure 2. Cost calculator 216 includes fixed disk cost multipliers 408-418, weighting multipliers 420-430, and a cost summer 432. Permutations 310-330 are respectively input to fixed disk cost multipliers 408-418. The value of each respective fixed disk multiplier is adjusted based on the disk's measured performance through feedback 128. The output of each respective fixed disk multiplier is weighted by the length of each corresponding disk queue for disks A, B, C, P, Q and R through 232a-f. The result for each permutation is calculated by summer 432 and output as permutation expense 436. The summer 432 may add the input results, or may

expense the largest input result, or may use another appropriate algorithm. If necessary, hinting information 234 is output to permutation generator 212 (Figures 2 and 3).

[25] Figure 5 is a functional block diagram showing greater detail of combiner 122, shown in Figure 1. Combiner 122 includes disk return queues 504-514, a permutation return buffer 516, a permutation collector 518, an actual cost calculator 520 and a reconstructed data return buffer 522. Each disk A, B, C, P, Q and R respectively returns data to disk return queues 504-514, which operate in a well-known manner. The selected permutation for a SOLD triple is output from permutation multiplexer 218 to permutation return buffer 516 as state update information 124. Permutation collector 518 receives a completed permutation in disk return queues 504-514 and the selected permutation from permutation return buffer 516. The actual cost of the permutation is then calculated by actual cost calculator 520. Actual cost information is returned to cost calculator 216 as feedback information 128 (Figures 2 and 3). The data for the original SOLD is reconstructed by reconstructed data return buffer 522 and output as return data 126 (Figure 1).

[26] For example, consider the costing of the situation wherein which disks A and B are moderately loaded and disk R has failed; and in which the SOLD requests one sector from disk A. Further, the metric in this example uses a trivial estimate of the sum of queue times for retrieving the data as the cost. Assume that disk A queue 220 has a length of 40 outstanding requests, disk B queue 222 has a length of 35, disk C queue 224 has a length of 5, disk P queue 226 has a length of 8, disk Q queue 228 has a length of 7 and disk R queue 230 has an indeterminate length. These queue lengths are passed to the cost calculator 216 through pathways 232a-f. Cost calculator 216 knows that requests to identical disks A, B, C, P and Q take 10 ms to complete and

these weights are stored in fixed disk cost multipliers 408-416. Disk R, having failed, is taking more than 10^6 s to complete its requests 418. Hinting pathway 234 informs the permutation generator 212 that disk R has failed. Thus, when the SOLD request for disk A is received 302, the permutation generator 212 generates pathway permutations on disk A 310, disks BCQ 314, disks BCP 320, disks CPQ 324 and disks BPQ 326. The metric for pathway 310 is 40 x 10 ms + 0 + 0 + 0 + 0 + 0 = 400 ms, and for pathway 314 the metric is 0 + 35 x 10 ms + 5 x 10 ms + 0 + 7 x 10 ms = 470 ms. Similarly, pathway 320 has a summed cost of 480 ms, pathway 324 has a cost of 200 ms and pathway 326 has a cost of 500 ms. Thus, the lowest cost path is pathway 324 issuing reads to disks C, P and Q. The chosen pathway is conveyed to combiner 122 through state update 124. Combiner 122 receives the relevant data from disks C, P and Q for reconstructing disk A's information in the permutation collector 518. The measured times to complete the request is collated in 520 and the results used to update cost calculator 216 through feedback 128. In this example, the metric is very simple and the working disks are identical in performance so feedback is not required. A more sophisticated cost calculator, however, may, for example, account for the fact that the time for a disk to complete a request is not linear in the queue length, as is assumed here. The feedback mechanism permits the cost calculator to use a more accurate estimate.

[27] The present invention can determine an optimal pathway for retrieving data from a data storage system as requests for data are received. Alternatively, the present invention determines optimal pathways in advance, based on current operating conditions of the storage system, and selects a predetermined optimal pathway for each received data request. Moreover, the present invention can be configured as a processing system within a storage system that executes machine-language

instructions or as individual components contained within a storage system that perform the respective functions of the present invention.

[28] While the present invention has been described with respect to a specific example illustrating a general process for selecting a best pathway to data and for improving performance of data reads of a storage system, those skilled in the art will appreciate that there are numerous variations and permutations of the above described systems and techniques that fall within the spirit and scope of the invention as set forth in the appended claims.